Computational Aspects of High-Resolution Global Gravity Field Determination – Numbering Schemes and Reordering

J. M. Brockmann, W.-D. Schuh

published in

NIC Symposium 2016
K. Binder, M. Müller, M. Kremer, A. Schnurpfeil (Editors)

Forschungszentrum Jülich GmbH,
John von Neumann Institute for Computing (NIC),
Schriften des Forschungszentrums Jülich, NIC Series, Vol. 48,
ISBN 978-3-95806-109-5, pp. 309.
http://hdl.handle.net/2128/9842

Computational Aspects of High-Resolution Global Gravity Field Determination – Numbering Schemes and Reordering

Jan Martin Brockmann and Wolf-Dieter Schuh

Institute of Geodesy and Geoinformation, Department of Theoretical Geodesy,
University of Bonn, Nussallee 17, D-53115 Bonn, Germany
E-mail: {brockmann, schuh}@geod.uni-bonn.de

Estimating high-degree spherical harmonic gravity field models from complementary observation types is computationally demanding. The computational effort depends on the one hand on the maximal resolution of the spherical harmonic expansion (i.e. the number of parameters to be estimated, tens to hundreds of thousands) and on the other hand on the number of observations (up to hundreds of millions). If approximations (e.g. block-diagonal approximations) should be avoided, concepts of high-performance computing have to be used to compute rigorous least-squares solutions from the typically dense systems of equations.

Within this contribution we focus on a technical detail in the context of estimating combined global gravity field models. Typically, dense systems of normal equations of different resolutions and provided in different numbering schemes have to be combined. Due to their dimension, they are mapped to the main memory as block-cyclically distributed matrices. We introduce symbolic numbering schemes, which are used to describe the resolution and the parameter order. This contribution summarises, how these symbolic numbering schemes are used to determine the permutation between the parameter order of two normal equations of different resolution and how the reordering is performed.

1 Introduction and Motivation

Gravity field models of the Earth are highly relevant in many geo-scientific applications[1]. With a growing database (through dedicated satellite gravity missions like GRACE[2] or GOCE[3], altimetry or surface data) combined models with higher spatial resolution become possible. The state-of-the-art mathematical parameterisation to describe the global Earth's gravitational potential is a spherical harmonic expansion up to a certain maximal degree $l_{\max}$. The potential may be written, for some evaluation point (r, θ, λ) in an Earth fixed and centred coordinate system, as[4]

$$V(r,\theta,\lambda) = \frac{GM}{a} \sum_{l=0}^{l_{\max}} \left(\frac{a}{r}\right)^{l+1} \sum_{m=0}^{l} \left(c_{lm} \cos(m\lambda) + s_{lm} \sin(m\lambda)\right) P_{lm}(cos\theta), \quad (1)$$

where l and m denote the spherical harmonic degree and order (d/o), c_{lm} and s_{lm} the unknown coefficients of the spherical harmonic expansion, a the equatorial radius of the Earth reference ellipsoid, $P_{lm}(\cdot)$ the fully normalised associated Legendre functions, and GM the geocentric gravitational constant. This model comprises $U = (l_{\max}+1)^2$ unknown coefficients, which can be used to express related geometrical and physical functionals of the Earth's gravity field.

Within global gravity field determination, the unknown coefficients c_{lm} and s_{lm} are estimated in a least-squares adjustment from various data sources, i.e. observations of the

effect of the Earth's gravity field (gravity changes, satellite orbit disturbances, accelerations, ...)[1,4,5]. These observations are either globally collected by satellites or locally in terrestrial measurement campaigns. Depending on the used data sets and their characteristics tens to several hundreds of thousands of parameters have to be estimated from the data. The unknown parameters from multiple data sources indicated by subscripts o and n are typically estimated in a joint weighted least squares adjustment[5,6]. The parameters $\mathbf{x}$ are determined via the assembly and solution of the combined normal equations[7–9] (NEQs)

$$\left(\sum_o \omega_{\mathrm{o}} \mathbf{A}_{\mathrm{o}}^T \boldsymbol{\Sigma}_{oo} \mathbf{A}_{\mathrm{o}} + \sum_n \omega_n \mathbf{N}_n\right) \mathbf{x} = \left(\sum_o \omega_{\mathrm{o}} \mathbf{A}_{\mathrm{o}}^T \boldsymbol{\Sigma}_{oo} \boldsymbol{\ell}_{\mathrm{o}} + \sum_n \omega_n \mathbf{n}_n\right). \quad (2)$$

The observation groups o are available as raw measurements with design matrices $\mathbf{A}_o$, stochastic data vectors $\boldsymbol{\ell}_o$, and covariance matrices $\boldsymbol{\Sigma}_{oo}$ describing the uncertainty characteristics of the observation vector. The observation groups n are assumed to be already available as NEQs, where $\mathbf{N}_n$ are the preprocessed NEQ matrices and $\mathbf{n}_n$ the right hand side vectors. $\omega_{n,o}$ are (unknown) weight factors for the observation groups. The vector $\mathbf{x}$ contains the unknown parameters, i.e. mainly spherical harmonic coefficients c_{lm} and s_{lm} but also additional group specific parameters like biases, or finite element parameters if the dynamic ocean topography is co-estimated from the altimeter data[10,11].

Although it is generally useful to separate the group n and o as they have to be treated separately within the entire procedure[9], they can be merged within this contribution. Defining $\mathbf{N}_o := \mathbf{A}_{\mathrm{o}}^T \boldsymbol{\Sigma}_{oo} \mathbf{A}_{\mathrm{o}}$ and $\mathbf{n}_o := \mathbf{A}_{\mathrm{o}}^T \boldsymbol{\Sigma}_{oo} \mathrm{l}_{\mathrm{o}}$, which assumes the direct computations of the NEQs from groups o, Eq. 2 can be re-written with a single index $i \in \{o, n\}$ as

$$\left(\sum_i \omega_i \mathbf{N}_i\right) \mathbf{x} = \sum_i \omega_i \mathbf{n}_i, \Leftrightarrow \mathbf{N}\mathbf{x} \quad = \mathbf{n}, \text{ with } \mathbf{N} := \sum_i \omega_i \mathbf{N}_i, \mathbf{n} := \sum_i \omega_i \mathbf{n}_i. \quad (3)$$

Due to the dimension of tens to hundreds of thousands of rows and columns[9] of the symmetric and positive definite system of NEQs (Eq. 3), their setup and solution is implemented via an integrated use of block-cyclically distributed matrices[12,9,8] and SCALAPACK. Although the task of combining the group specific NEQs $\mathbf{N}_i$ and $\mathbf{n}_i$ seems simple, Eq. 3 is only valid if the observation equations and thus the NEQs of all groups i are all assembled for the same and entire target parameter space (in the same parameter order).

Within gravity field determination, this is typically not the case as: (i) the NEQs from individual data sets are typically set up only up to a certain spherical harmonic degree, i.e. a degree determined by the sensitivity of of the measurement concept. For instance this can be $l_{\max} = 5$ for Satellite Laser Ranging[13] but $l_{\max} = 280$ for GOCE[14]. (ii) The NEQs may contain group specific parameters, e.g. biases, calibration parameters or additional target parameters which can be determined from specific observation groups only[10]. Finally, (iii) the ordering of the parameters is varying, as it is more or less arbitrary, although different (standard) so called numbering schemes with different properties exist[15]. If an arbitrary number of NEQs should be combined, online reordering is required to guarantee consistent combined NEQs.

This contribution focuses on the combination of different NEQs assuming the parameter space and ordering of the different groups i differs. A reordering strategy is developed to derive a general and flexible framework. Based on a symbolic description of the parameter ordering (numbering scheme) efficient reordering schemes can be applied. As within

global gravity field determination the NEQs are high-dimensional, the concept is applied to block-cyclically distributed matrices. The developed framework is usable for many applications and in different situations, e.g. prerequisites for different solvers[9,16].

2 Reordering of Block-Cyclically Distributed Matrices

2.1 Symbolic Numbering Schemes

Instead of relying on rule-defined numbering schemes[17,15] it is more flexible and more general to define symbolic numbering schemes. A symbolic numbering scheme can be defined as a linear sequence of symbolically described parameters, for instance objects of a type `Parameter`. The linear sequence can be realised as a `std::vector<Parameter>`. The class `Parameter` and their attributes are used to uniquely describe the parameters, the parameter type (e.g. spherical harmonic coefficient, fine element coefficient, ...) plus additional information like d/o for a spherical harmonic coefficient. Every parameter group can be mapped to this scheme. For the later described determination of the reordering operation between two symbolic numbering schemes, it is essential that parameters are comparable such that they are sortable. As every parameter in a numbering scheme is unique, it is simple to define a `operator<(const Parameter & p)`. A numbering scheme, a sequence of parameters is called $\mathbf{p}$ in the following, whereas p is an individual parameter and $\mathbf{p}[i]$ the ith parameter in the numbering scheme. Each numbering scheme, either generated based on rules or arbitrarily chosen, can be represented by such a symbolic numbering scheme. Thus, it is easy to assume that for every NEQ used in the combination, a symbolic numbering scheme can be created online (rule-based) or is available (from a file).

2.2 Reordering of Matrices, Index and Permutation Vectors

Based on two numbering schemes $\mathbf{p}_f$ and $\mathbf{p}_t$, the goal is to find the index vector and the permutation operator which reorders a vector $\mathbf{x}_{\mathbf{p}_f}$ given in $\mathbf{p}_f$ to the vector $\mathbf{x}_{\mathbf{p}_t}$ which contains the parameters sorted as described by $\mathbf{p}_t$. The requirements for $\mathbf{p}_f$ and $\mathbf{p}_t$ are that either all coefficients of $\mathbf{p}_f$ are contained in $\mathbf{p}_t$ or vise versa, that all coefficients of $\mathbf{p}_t$ are contained in $\mathbf{p}_f$. The first case means that $\mathbf{p}_f$ contains fewer parameters then $\mathbf{p}_t$, i.e. $\mathbf{p}_f$ is a subset of $\mathbf{p}_t$. The vector to be reordered shall be extended by zeros, corresponding to entries of "missing" coefficients. In the other case, $\mathbf{p}_t$ contains fewer parameters then $\mathbf{p}_f$, consequently $\mathbf{p}_t$ is a subset of $\mathbf{p}_f$. In that situation, the vector should be reordered in such a way that the first coefficients correspond to the ordering of $\mathbf{p}_t$ and that the additional coefficients are ordered to the end of the vector, which can then be truncated. The algorithms provided are operational for both cases. Nevertheless, for the combination of NEQs focused on here, the first case is the crucial one. The change of the parameter order (reordering) is nothing else than an interchange of rows for vectors (like the right hand side $\mathbf{n}_i$) and the interchange of rows and columns for matrices (like the normal matrices $\mathbf{N}_i$). Mathematically the operation can be described by an index vector $\mathbf{i}_{\mathbf{p}_f \mapsto \mathbf{p}_t}$ or a permutation operation $\mathbf{\Psi}_{\mathbf{p}_f \mapsto \mathbf{p}_t}$. Whereas the index vector assumes a simultaneous interchange, the permutation assumes a sequential interchange (taking already performed interchanges into account).

Algorithm 1: Computation of index vector from two symbolic numbering schemes.

```
   Data:  vector<Parameter> p_from  Symbolic numbering scheme source matrix is ordered in
          vector<Parameter> p_into  Symbolic numbering scheme matrix should be reordered to
 1 vector<size_t> i_{p_from↦p_into}( p_into.size(), 0 )        // initialisation of index vector
 2 // start value for fill in indices for parameters in p_into but not in p_from, inserted at the end
 3 size_t e = p_from.size()
 4 // store current index of parameter in auxiliary variable _i of each individual parameter
 5 for k = 0 to p_from.size() do
 6 |     p_from[k].i() = k
 7 end
 8 sort(p_from.begin(), p_from.end())            // sort numbering scheme (operator<)
 9 // loop over all parameters in p_into
10 for k = 0 to p_into.size() do
11 |     // find index of parameter p_into(i) in p_from in sorted numbering scheme
12 |     i = lower_bound(p_from.begin(), p_from.end(), p_into(i) )
13 |     // if parameter found, insert index i, otherwise fill in value outside of p_from.size()
14 |     if i< p_from.size() then
15 |     |     i_{p_from↦p_into}(k) = p_from.p(i).i()
16 |     else
17 |     |     i_{p_from↦p_into}(k) = e
18 |     |     e++
19 |     end
20 end
21
22 // special case if p_into ⊆ p_from: extend index vector to size of p_from
23 if p_into.size() < p_from.size() then
24 |     i_{p_from↦p_into}.resize(p_from.size())
25 |     // the remaining parameters are sorted to the end as they are not contained in p_into,
26 |     for k = p_into.size() to p_from.size() do
27 |     |     i_{p_from↦p_into}(k) = k
28 |     end
29 end
30 return i_{p_from↦p_into} // index vector performing reordering from p_from to p_into
```

Index Vector for Reordering

Within the index vector $\mathbf{i}_{\mathbf{p}_\mathrm{f}\mapsto\mathbf{p}_\mathrm{t}}$, the entry at position i contains the index, the coefficient $\mathbf{p}_\mathrm{t}(i)$ can be found in $\mathbf{p}_\mathrm{f}$, and thus $\mathbf{p}_\mathrm{t}(i) = \mathbf{p}_\mathrm{f}(\mathbf{i}_{\mathbf{p}_\mathrm{f}\mapsto\mathbf{p}_\mathrm{t}}(i))$. Given a matrix $\mathbf{A}$ in $\mathbf{p}_\mathrm{f}$, its rows are reordered to $\mathbf{p}_\mathrm{t}$ via $\mathbf{A}_{\mathbf{p}_\mathrm{t}} = \mathbf{A}_{\mathbf{p}_\mathrm{f}}(\mathbf{i}_{\mathbf{p}_\mathrm{f}\mapsto\mathbf{p}_\mathrm{t}}, :)$, its columns via $\mathbf{A}_{\mathbf{p}_\mathrm{t}} = \mathbf{A}_{\mathbf{p}_\mathrm{f}}(:, \mathbf{i}_{\mathbf{p}_\mathrm{f}\mapsto\mathbf{p}_\mathrm{t}})$, and for quadratic matrices rows and columns via $\mathbf{A}_{\mathbf{p}_\mathrm{t}} = \mathbf{A}_{\mathbf{p}_\mathrm{f}}(\mathbf{i}_{\mathbf{p}_\mathrm{f}\mapsto\mathbf{p}_\mathrm{t}}, \mathbf{i}_{\mathbf{p}_\mathrm{f}\mapsto\mathbf{p}_\mathrm{t}})$, using the well known MatLab/Octave like notation. The index vector $\mathbf{i}_{\mathbf{p}_\mathrm{f}\mapsto\mathbf{p}_\mathrm{t}}$ can be computed with the efficient Alg. 1, which works for both cases mentioned above.

Within each coefficient of $\mathbf{p}_\mathrm{f}$, the original position in the numbering scheme is stored (cf. Alg. 1, l. 5–7) in a auxiliary attribute `i`. Afterwards $\mathbf{p}_\mathrm{f}$ can be sorted. Now, iterating over the parameters in $\mathbf{p}_\mathrm{t}$, they can be efficiently found, if contained, in $\mathbf{p}_\mathrm{f}$ as they have to be searched in a sorted vector (cf. Alg. 1, l. 10–20). The original index is stored from the auxiliary variable into the index vector. If the parameter is not contained in $\mathbf{p}_\mathrm{f}$, the index is set to an entry larger than $\mathbf{p}_\mathrm{f}$.`size()`, which will correspond to the extended zeros when the reordering is performed to a vector/matrix. Parameters contained in $\mathbf{p}_\mathrm{f}$ but not in $\mathbf{p}_\mathrm{t}$ are arranged to the end, via setting the entries of the index vector to a value larger then the

Algorithm 2: Conversion of an index vector to a permutation vector.

Data: `vector<size_t>` $\mathbf{i}_{\mathbf{p}_{\text{from}} \mapsto \mathbf{p}_{\text{into}}}$ index vector to be converted to permutation vector

`vector<size_t>` $\boldsymbol{\psi}_{\mathbf{p}_{\text{from}} \mapsto \mathbf{p}_{\text{into}}} = \mathbf{i}_{\mathbf{p}_{\text{from}} \mapsto \mathbf{p}_{\text{into}}}$ *// initialisation of permutation vector*

`size_t` $p = 0$

// auxiliary vector, entry $\mathbf{h}(k)$ *contains index where value* k *is stored in* $\boldsymbol{\psi}_{\mathbf{p}_{from} \mapsto \mathbf{p}_{into}}$

`vector<size_t>` $\mathbf{h}$ ($\mathbf{i}_{\mathbf{p}_{\text{from}} \mapsto \mathbf{p}_{\text{into}}}$ `.size()` , 0)

for $k = 0$ **to** $\mathbf{i}_{\mathbf{p}_{from} \mapsto \mathbf{p}_{into}}$ `.size()` **do**

  $\mathbf{h}(\mathbf{i}_{\mathbf{p}_{\text{from}} \mapsto \mathbf{p}_{\text{into}}}(k)) = k$

end

// loop over entries of index vector

for $k = 0$ **to** $\mathbf{i}_{\mathbf{p}_{from} \mapsto \mathbf{p}_{into}}$ `.size()` **do**

  $p = \mathbf{h}(k)$ *// index of number* k *follows from* $\mathbf{h}$ *instead of* `find` *operation*

  // check if entry k *is in subsequent part of vector*

  if $p > k$ **then**

    $\boldsymbol{\psi}_{\mathbf{p}_{\text{from}} \mapsto \mathbf{p}_{\text{into}}}(p) = \boldsymbol{\psi}_{\mathbf{p}_{\text{from}} \mapsto \mathbf{p}_{\text{into}}}(k)$

    $\mathbf{h}(\boldsymbol{\psi}_{\mathbf{p}_{\text{from}} \mapsto \mathbf{p}_{\text{into}}}(k)) = p$ *// update vector* $\mathbf{h}$*, value* $\boldsymbol{\psi}_{\mathbf{p}_{from} \mapsto \mathbf{p}_{into}}(k)$ *is now at position* p

  end

end

return $\boldsymbol{\psi}_{\mathbf{p}_{\text{from}} \mapsto \mathbf{p}_{\text{into}}}$ *// sequential permutation vector corresponding to* $\mathbf{i}_{\mathbf{p}_{\text{from}} \mapsto \mathbf{p}_{\text{into}}}$

dimension of $\mathbf{p}_{\text{t}}$ (cf. Alg. 1, l. 23–29). The complexity of the algorithm is $\mathcal{O}(n \log(n))$. For a test case of two numbering schemes of $520\,000$ parameters, the serial runtime is 0.2 s.

Permutation Vector for Reordering

An alternative notation/operation, which is better suited for the block-cyclically distributed matrices, is a so called permutation or pivoting vector. In contrast to an index vector, where the column and/or row interchanges are assumed to be performed simultaneously, a permutation vector contains a sequence of serial row and column permutations, i.e. a sequential swapping of two rows/columns starting at the begin of the vector. In contrast to the index vector, already performed swapping operations are taken into account in the representation. An entry in the permutation vector at position i means that the row i is swapped with row $\boldsymbol{\psi}(i)$. To be more precise, the current content of row/column i is swapped with the current content of row/column $\boldsymbol{\psi}(i)$. Note that the content might change with every swapping operation. Now, the old entry of position i is in row $\boldsymbol{\psi}(i)$, thus the index vector needs to be updated. A remaining entry i in the subsequent elements of $\mathbf{i}_{\mathbf{p}_{\text{f}} \mapsto \mathbf{p}_{\text{t}}}$ has to be replaced by the entry $\boldsymbol{\psi}(i)$. The procedure to convert an index vector to a permutation vector is summarised in Alg. 2. The basic idea is to avoid the search operation via introducing a second vector which stores the position of an entry k in the vector. The complexity is $\mathcal{O}(2n)$, its serial runtime is 0.01 s, for an example index vector with $520\,000$ entries.

To apply a permutation vector to rows and/or columns, the operator $\boldsymbol{\Psi}_{\mathbf{p}_{\text{f}} \mapsto \mathbf{p}_{\text{t}}}(\cdot)$ is defined. This operator performs the serial permutations of rows ($\boldsymbol{\Psi}^{r}_{\mathbf{p}_{\text{f}} \mapsto \mathbf{p}_{\text{t}}}$) as given by the vector $\boldsymbol{\psi}_{\mathbf{p}_{\text{f}} \mapsto \mathbf{p}_{\text{t}}}$, the operator $\boldsymbol{\Psi}^{c}_{\mathbf{p}_{\text{f}} \mapsto \mathbf{p}_{\text{t}}}$ performs the column interchanges and $\boldsymbol{\Psi}^{r,c}_{\mathbf{p}_{\text{f}} \mapsto \mathbf{p}_{\text{t}}}$ performs the interchanges for rows and columns.

3 Combined System of NEQs

Assuming that a target numbering scheme $\mathbf{p}$ (associated with $\mathbf{N}$) exists which covers the entire parameter space to be estimated, Eq. 3 can be rewritten as

$$\left(\sum_i w_i \begin{bmatrix} \mathbf{N}_i & \mathbf{0}_{U_i \times U-U_i} \\ \mathbf{0}_{U-U_i \times U_i} & \mathbf{0}_{U-U_i \times U-U_i} \end{bmatrix} (\mathbf{i}_{\mathbf{p}_i \mapsto \mathbf{p}}, \mathbf{i}_{\mathbf{p}_i \mapsto \mathbf{p}})\right) \mathbf{x} = \sum_i w_i \begin{bmatrix} \mathbf{n}_i \\ \mathbf{0}_{U-U_i} \end{bmatrix} (\mathbf{i}_{\mathbf{p}_i \mapsto \mathbf{p}}), \quad (4)$$

for the use with an index vector. Using the introduced permutation operator, we obtain

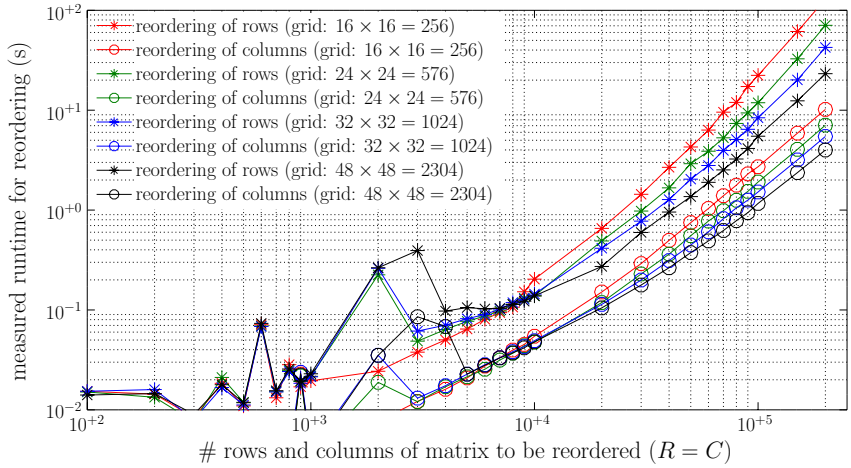
$$\left(\sum_i w_i \mathbf{\Psi}^{r,c}_{\mathbf{p}_i \mapsto \mathbf{p}} \left(\begin{bmatrix} \mathbf{N}_i & \mathbf{0}_{U_i \times U-U_i} \\ \mathbf{0}_{U-U_i \times U_i} & \mathbf{0}_{U-U_i \times U-U_i} \end{bmatrix}\right)\right) \mathbf{x} = \sum_i w_i \mathbf{\Psi}^{r}_{\mathbf{p}_i \mapsto \mathbf{p}} \left(\begin{bmatrix} \mathbf{n}_i \\ \mathbf{0}_{U-U_i} \end{bmatrix}\right), \quad (5)$$

assuming $\mathbf{N}_i$ and $\mathbf{n}_i$ to be the original NEQs as they were originally set up. Their numbering scheme is denoted as $\mathbf{p}_i$. These NEQs for the subset of the parameters are extended with zeros, if required. Afterwards the index vector or the permutation vector is applied to the temporarily extended NEQs. The NEQs can be combined performing a simple addition as the parameter order and parameter space is adjusted to the defined target numbering scheme $\mathbf{p}$. The solution of the NEQs equations can be performed with SCALAPACK[8,9].

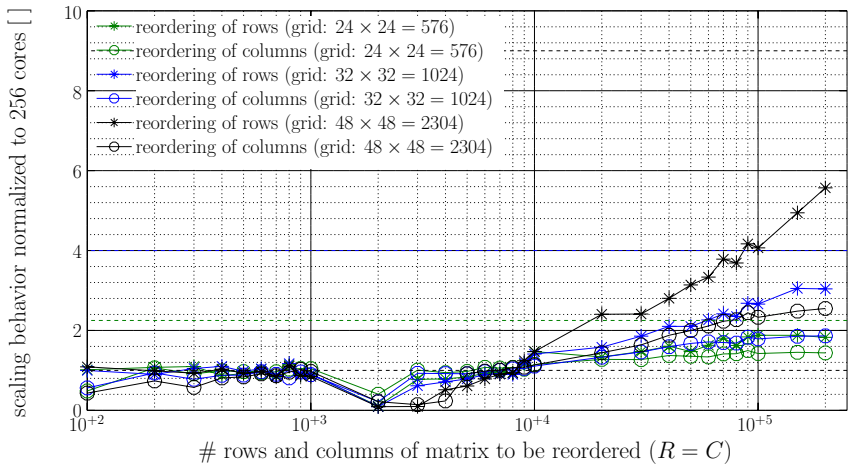
4 Reordering for Block-Cyclically Distributed Matrices

With the known permutation vector between two numbering schemes, the permutation is applied to block-cyclically distributed matrices to compute the combined system of NEQs cf. Eq. 5. With the permutation vector, the operation is simply performed by the SCALAPACK helper routine `pdlapiv`, which is used within SCALAPACK for pivoting during the solution of systems of equations. The subroutine performs the permutation given by $\boldsymbol{\psi}_{\mathbf{p}_f \mapsto \mathbf{p}_t}$ to either rows or columns of a block-cyclically distributed matrix. Applying the function twice, first to permute rows and secondly to permute columns, both are reordered. Beside the standard input of the block-cyclic distribution of the matrix, the function requires the input of the permutation vector as block-cyclically distributed integer vector. Note that before reordering, a symmetric matrix has to be stored in the lower and upper triangle, as during reordering, both are mixed.

Fig. 1 gives an overview of the required runtime for the reordering of rows and columns of distributed matrices of different dimension on different quadratic processor grids (for the distribution parameters default values of $b_r = b_c = 64$ were used) to get an idea of the order of magnitude. The index vector was randomly generated (random shuffle of an index vector). The main conclusions of the test are: i) The reordering of columns is much faster than the reordering of rows (by a factor of three to ten). This could be expected as the column access in memory is much faster using the column major order for matrices for the locally stored matrices. ii) For matrices of dimension lower than $20\,000 \times 20\,000$ the reordering is performed in less than 1 s on all grids. For the reordering of columns, this even holds for matrices smaller than $80\,000 \times 80\,000$. Although there is no real scaling behaviour of the reordering operations with the number of cores (cf. Fig. 1(b)), the most important thing is that the performance increases on larger compute core grids and does not drop to additional organisational requirements (at least for matrices above dimension $10\,000 \times 10\,000$ the scaling is above 1.0 for all cases analysed).



(a) Mean absolute runtime measured for the reordering.



(b) Scaling behaviour of reordering operations (normalised to 256 cores).

Figure 1. Runtime analysis of the row (⋆) and column (○) reordering operations (index vector randomly generated). The colours represent different dimensions of the processor grid.

5 Summary and Conclusions

The concept of symbolic numbering schemes and the strategy to reorder block-cyclically distributed matrices is used within the framework for global gravity field recovery[9]. The reordering is computed efficiently on the fly and the original NEQs can be kept, and no additional preprocessing and homogenisation of the NEQs into a predefined numbering scheme is necessary. These strategies are also required within global gravity determination using iterative solvers, where the NEQs are required in different numbering schemes for the efficient setup of the observation equations (recursion formulas) and for preconditioning (block diagonal dominance in specific numbering)[9,16].

Acknowledgements

Parts of the study and the publication were funded within the DFG project G/O2000+ (SCHU2305/3-1). Most computations within the (entire) study were performed on the JUROPA supercomputer at Forschungszentrum Jülich. The computing time was granted by the John von Neumann Institute for Computing (project HBN15). The Open Source HPC computing libraries ATLAS, BLAS, LAPACK, OpenMPI, BLACS, PBLAS and SCALAPACK are gratefully acknowledged.

References

1. R. Rummel, G. Balmino, J. Johannessen, P.N.A.M. Visser, and P. Woodworth, *Dedicated gravity field missions – principles and aims*, J Geodyn, **33**, no. 12, 2002.
2. B. D. Tapley, S. Bettadpur, M. Watkins, and C. Reigber, *The gravity recovery and climate experiment: Mission overview and early results*, Geophys Res Lett, **31**, 2004.
3. R. Floberghagen, M. Fehringer, D. Lamarre, D. Muzi, B. Frommknecht, C. Steiger, J. Pineiro, and A. da Costa, *Mission design, operation and exploitation of the gravity field and steady-state ocean circulation explorer mission*, J Geodesy, **85**, no. 11, 749–758, 2011.
4. B. Hofmann-Wellenhof and H. Moritz, *Physical geodesy*, Springer, Vienna, 2005.
5. N. K. Pavlis, S. A. Holmes, S. Kenyon, and J. K. Factor, *The development and evaluation of the Earth Gravitational Model 2008 (EGM2008)*, J Geophys Res: Solid Earth, **117**, no. B4, 2012.
6. T. Gruber, *High resolution gravity field modeling with full variance-covariance matrices*, J Geodesy, **75**, no. 9-10, 505–514, 2001.
7. K. R. Koch, *Parameter Estimation and Hypothesis Testing in Linear Models*, Springer Berlin Heidelberg, 2nd edition, 1999.
8. J. M. Brockmann, L. Roese-Koerner, and W.-D. Schuh, *Use of high performance computing for the rigorous estimation of very high degree spherical harmonic gravity field models*, in: Procceedings of the International Symposium on Gravity, Geoid and Height Systems (GGHS2012), U. Marti (Ed.), vol. 141 of *International Association of Geodesy Symposia*, pp. 27–33, Springer Berlin Heidelberg. 2015.
9. J. M. Brockmann, *On High Performance Computing in Geodesy – Applications in Global Gravity Field Determination*, PhD thesis, Institute of Geodesy and Geoinformation, University of Bonn, Bonn, Germany, 2014.
10. S. Becker, M. Losch, J. M. Brockmann, G. Freiwald, and W.-D. Schuh, *A Tailored Computation of the Mean Dynamic Topography for a Consistent Integration into Ocean Circulation Models*, Surv Geophys, **35**, no. 6, 1507–1525, 2013.
11. S. Becker, J. M. Brockmann, and W.-D. Schuh, *Mean dynamic topography estimates purely based on GOCE gravity field models and altimetry*, Geophys Res Lett, **41**, no. 6, 2063–2069, 2014.
12. L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley, *ScaLAPCK Users Guide*, SIAM, Society for Industrial and Applied Mathematics, Philadelphia, 2nd edition, 1997.

13. A. Maier, S. Krauss, W. Hausleitner, and O. Baur, *Contribution of satellite laser ranging to combined gravity field models*, Adv Space Res, **49**, no. 3, 556–565, 2012.
14. J. M. Brockmann, N. Zehentner, E. Höck, R. Pail, I. Loth, T. Mayer-Gürr, and W.-D. Schuh, *EGM_TIM_RL05: An independent Geoid with Centimeter Accuracy purely based on the GOCE Mission*, Geophys Res Lett, **41**, no. 22, 8089–8099, 2014.
15. C. Boxhammer and W.-D. Schuh, *GOCE gravity field modeling: computational aspects - free kite numbering scheme*, in: Observation of the Earth System from Space, J. Flury et al. (Eds.), pp. 209–224. Springer Berlin Heidelberg, 2006.
16. J. M. Brockmann, L. Roese-Koerner, and W.-D. Schuh, *A concept for the estimation of high-degree gravity field models in a high performance computing environment*, Stud Geophys Geod, **58**, 571–594, 2014.
17. W.-D. Schuh, *Tailored Numerical Solution Strategies for the Global Determination of the Earth's Gravity Field*, vol. 81 of *Mitteilungen der geodätischen Institute der Technischen Universität Graz*, TU Graz, Graz, Austria, 1996.